**SECURE SHARE**



***A project report submitted to***

***Rajiv Gandhi Proudyogiki Vishwavidhyalaya, Bhopal***

***in partial fulfillment for the award of***

***the degree of***

***Bachelor of Engineering***

***in***

***Computer Science & Engineering***

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

SRI AUROBINDO INSTITUTE OF TECHNOLOGY

**INDORE- 452001**

**2020**

**SECURE SHARE**



***A project report submitted to***

***Rajiv Gandhi Proudyogiki Vishwavidhyalaya, Bhopal***

***in partial fulfillment for the award of***

***the degree of***

***Bachelor of Engineering***

***in***

***Computer Science & Engineering***

**PROJECT GUIDE SUBMITTED BY**

DR.NITHIKA VATS DOOHAN Devesh Dhoble (0873cs181016)

Aditya Saxena (0873cs181007)

Gaurav Tripathi(0873cs181019)

Shubham Singh Chouhan

(0873cs181066)

**DEPARTMENT OF COMPUTER**

**SCIENCE & ENGINEERING**

SRI AUROBINDO INSTITUTE OF TECHNOLOGY

**INDORE- 452001**

**2020 – 2021**

SRI AUROBINDO INSTITUTE OF TECHNOLOGY

**INDORE- 452001**



**CERTIFICATE**

This is to certify that **Devesh Dhoble (0873cs181016), Aditya Saxena (0873cs181007), Gaurav Tripathi(0873cs181019), Shubham Singh Chouhan (0873cs181066)** have completed their project work, titled **“SECURE SHARE”**  as per the syllabus and have submitted a satisfactory report on this project as a part of fulfillment towards the degree of **“BACHELOR OF ENGINEERING” (Computer Science & Engineering)**  from **RAJIV GANDHI PROUDYOGIKI VISHWAVIDHYALAYA, BHOPAL.**

**HEAD OF THE DEPARTMENT PROJECT GUIDE**

**DIRECTOR**

SRI AUROBINDO INSTITUTE OF TECHNOLOGY

**INDORE- 452001**



**CERTIFICATE**

This is to certify that **Devesh Dhoble (0873cs181016), Aditya Saxena (0873cs181007), Gaurav Tripathi (0873cs181019), Shubham Singh Chouhan (0873cs181066)** have completed their project work, titled **“SECURE SHARE”** as per the syllabus and have submitted a satisfactory report on this project as a part of fulfillment towards the degree of **“BACHELOR OF ENGINEERING” (Computer Science & Engineering)**  from **RAJIV GANDHI PROUDYOGIKI VISHWAVIDHYALAYA, BHOPAL.**

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

We respect and thank **Mrs. NITIKA VATS DOOHAN**, for providing me an opportunity to do the project work in Sri Aurobindo Institute of Technology  and giving us all  the support and guidance which made me complete the project duly. I am extremely thankful to him for providing such a nice support and guidance, although he had a busy schedule managing the Department.

We owe my deep gratitude to our project guide **Mrs. NITIKA VATS DOOHAN**. They took a keen interest in our project work and guided us all along, till the completion of our project work by providing all the necessary information for developing a good system.

We would not forget to remember my teammates, project  for their help and more over for their timely support and guidance till the completion of our project work.

We heartily thank our internal project guide, **Mrs. NITIKA VATS DOOHAN**, Professor, Computer Science  and  Engineering  Department for his guidance and suggestions during this project work.

We are thankful and fortunate enough to get constant encouragement, support and guidance from all Teaching staff of the Computer Science and Engineering Department which helped us in successfully completing our project work. Also, I would like to extend our sincere esteem to all staff in the laboratory for their timely support.

**STUDENTS NAME with EnrNo**

Devesh Dhoble (0873cs181016)

Aditya Saxena (0873cs181007)

Gaurav Tripathi(0873cs181019)

Shubham Singh Chouhan

(0873cs181066)

ABSTRACT

This document is meant for describing all the features and procedures that were followed while developing the project. This document specially mentions the details of the project how it was developed, the primary requirement as well as various features and functionalities of the project and the procedures followed in achieving the objectives. Secure share is a Program to secure files (files of all extensions) and folder with user desired password as well as for sharing files over the network (peer to peer network), using cryptography.

**TABLE OF CONTENT**

**1: INTRODUCTION**

**2: PROBLEM STATEMENT**

**3: HARDWARE AND SOFTWARE REQUIREMENT**

**4: SOFTWARE DEVELOPMENT**

**5: DESIGN FRAMEWORK**

**6: IMPLEMENTATION**

**7: LIMITATION CONCLUSION AND FUTURE SCOPE**

**8: REFERENCES**

**INTRODUCTION**

Secure share is a program to secure files (files of all extensions) and folder with user desired password as well as provides you to security with your friends, employs etc also includes sharing files over and within the network (peer to peer network), using cryptography. The Program will work in windows operating system as well as Linux operating system.

## Cryptography is a method of protecting information and communications through the use of codes so that only those for whom the information is intended can read and process it.

Cryptography is a method of protecting information and communications through the use of codes, so that only those for whom the information is intended can read and process it. The prefix "crypt-" means "hidden" or "vault" -- and the suffix "-graphy" stands for "writing."

In computer science, cryptography refers to secure information and communication techniques derived from mathematical concepts and a set of rule-based calculations called algorithms, to transform messages in ways that are hard to decipher. These deterministic algorithms are used for cryptographic key generation, digital signing, verification to protect data privacy, web browsing on the internet, and confidential communications such as credit card transactions and email.

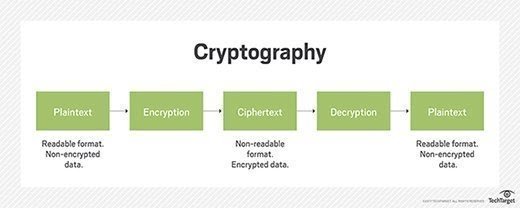
**Cryptography Techniques:**

Cryptography is closely related to the disciplines of [cryptology](https://searchsecurity.techtarget.com/definition/cryptology) and [cryptanalysis](https://searchsecurity.techtarget.com/definition/cryptanalysis). It includes techniques such as microdots, merging words with images, and other ways to hide information in storage or transit. However, in today's computer-centric world, cryptography is most often associated with scrambling [plaintext](https://searchsecurity.techtarget.com/definition/plaintext) (ordinary text, sometimes referred to as cleartext) into [ciphertext](https://whatis.techtarget.com/definition/ciphertext) (a process called [encryption](https://searchsecurity.techtarget.com/definition/encryption)), then back again (known as decryption). Individuals who practice this field are known as cryptographers.

Modern cryptography concerns itself with the following four objectives:

1. **Confidentiality**: the information cannot be understood by anyone for whom it was unintended
2. **Integrity:**the information cannot be altered in storage or transit between sender and intended receiver without the alteration being detected
3. **Non-repudiation**: the creator/sender of the information cannot deny at a later stage his or her intentions in the creation or transmission of the information
4. **Authentication**: the sender and receiver can confirm each other's identity and the origin/destination of the information

Procedures and [protocols](https://searchnetworking.techtarget.com/definition/protocol) that meet some or all of the above criteria are known as cryptosystems. Cryptosystems are often thought to refer only to mathematical procedures and computer programs; however, they also include the regulation of human behavior, such as choosing hard-to-guess passwords, logging off unused systems, and not discussing sensitive procedures with outsiders.



### Cryptographic algorithms

Cryptosystems use a set of procedures known as cryptographic algorithms, or ciphers, to encrypt and decrypt messages to secure communications among computer systems, devices such as smartphones, and applications. A cipher suite uses one algorithm for encryption, another algorithm for message authentication, and another for key exchange. This process, embedded in protocols and written in software that runs on operating systems and networked computer systems, involves public and private key generation for data encryption/decryption, digital signing and verification for [message authentication](https://searchsecurity.techtarget.com/definition/message-authentication-code-MAC), and key exchange.

### Types of cryptography

[Single-key or symmetric-key encryption](https://searchsecurity.techtarget.com/feature/Cryptography-basics-Symmetric-key-encryption-algorithms) algorithms create a fixed length of bits known as a [block cipher](https://searchsecurity.techtarget.com/definition/block-cipher) with a secret key that the creator/sender uses to encipher data (encryption) and the receiver uses to decipher it. Types of symmetric-key cryptography include the [Advanced Encryption Standard](https://searchsecurity.techtarget.com/definition/Advanced-Encryption-Standard) (AES), a specification established in November 2001 by the National Institute of Standards and Technology as a Federal Information Processing Standard (FIPS 197), to protect sensitive information. The standard is mandated by the U.S. government and widely used in the private sector.

In June 2003, AES was approved by the U.S. government for classified information. It is a royalty-free specification implemented in software and hardware worldwide. AES[is the successor to the Data Encryption Standard (DES)](https://searchsecurity.techtarget.com/answer/The-differences-between-AES-and-DES) and DES3. It uses longer key lengths (128-bit, 192-bit, 256-bit) to prevent brute force and other attacks.

[Public-key or asymmetric-key encryption](https://searchsecurity.techtarget.com/definition/asymmetric-cryptography) algorithms use a pair of keys, a public key associated with the creator/sender for encrypting messages and a private key that only the originator knows (unless it is exposed or they decide to share it) for decrypting that information. The types of public-key cryptography include [RSA](https://searchsecurity.techtarget.com/definition/RSA), used widely on the internet; Elliptic Curve Digital Signature Algorithm (ECDSA) used by Bit coin; Digital Signature Algorithm (DSA) adopted as a Federal Information Processing Standard for digital signatures by NIST in FIPS 186-4; and Diffie-Hellman key exchange.

To maintain data integrity in cryptography, hash functions, which return a deterministic output from an input value, are used to map data to a fixed data size. Types of cryptographic hash functions include SHA-1 (Secure Hash Algorithm 1), SHA-2 and SHA-3.

### Cryptography concerns

Attackers can bypass cryptography, hack into computers that are responsible for data encryption and decryption, and exploit weak implementations, such as the use of default keys. However, cryptography makes it harder for attackers to access messages and data protected by encryption algorithms.

Growing concerns about the processing power of quantum computing to break current cryptography encryption standards led the [National Institute of Standards and Technology (NIST)](https://searchsoftwarequality.techtarget.com/definition/NIST)to put out a call for papers among the mathematical and science community in 2016 for new public key cryptography standards. Unlike today's computer systems, quantum computing uses quantum bits (qubits) that can represent both 0s and 1s, and therefore perform two calculations at once. While a large-scale quantum computer may not be built in the next decade, the existing infrastructure requires standardization of publicly known and understood algorithms that offer a secure approach, according to [NIST](https://searchsoftwarequality.techtarget.com/definition/NIST). The deadline for submissions was in November 2017, analysis of the proposals is expected to take three to five years.

### History of cryptography:

The word "cryptography" is derived from the Greek kryptos, meaning hidden. The origin of cryptography is usually dated from about 2000 B.C., with the Egyptian practice of hieroglyphics. These consisted of complex pictograms, the full meaning of which was only known to an elite few. The first known use of a modern [cipher](https://searchsecurity.techtarget.com/definition/cipher) was by Julius Caesar (100 B.C. to 44 B.C.), who did not trust his messengers when communicating with his governors and officers. For this reason, he created a system in which each character in his messages was replaced by a character three positions ahead of it in the Roman alphabet.

In recent times, cryptography has turned into a battleground of some of the world's best mathematicians and computer scientists. The ability to securely store and transfer sensitive information has proved a critical factor in success in war and business.

Because governments do not wish certain entities in and out of their countries to have access to ways to receive and send hidden information that may be a threat to national interests, cryptography has been subject to various restrictions in many countries, ranging from limitations of the usage and export of software to the public dissemination of mathematical concepts that could be used to develop cryptosystems. However, the internet has allowed the spread of powerful programs and, more importantly, the underlying techniques of cryptography, so that today many of the most advanced cryptosystems and ideas are now in the public domain.

INDEPENDENT OF THE TYPE OF FILE:

There are various types of file types or extensions available to use. So we have made sure that any type of file format is workable in our application .

PEER TO PEER SHARING IS ENABLED WITH CRYPTOGRAPHY:

To increase the security features we have merged cryptography with peer to peer sharing in which every user will have his own public and private key.

SHARABLE FILES:

Since there are various software already available in the market which store user’s file in an encrypted manner. So we have made our software in such a manner so that if a user wants to share his files with his friends he can easily achieve this ( in a more secured manner).

SECURITY IS THE TOP PRIORITY

We have used a secured/standard algorithm (which encrypts our data) includes DES algorithm and AES algorithm.

**PROBLEM STATEMENT**

Sometimes user share system with another person in that case if user don’t want that another person could view or access his data from the drives, files or any particular folder in that case we are providing user to encrypt those files/ folder with his preferred password and file is only visible to him whenever user decrypt it with the same password.

Since there are various softwares already available in the market which store user’s file in an encrypted manner. But users are not always sure that is there data ( images, audio, video etc) is really secured over the server or transferred in encrypted form.

**Problem Statement**. Symmetric key **encryption** scheme with 40, 56, or 64 bit keys is subject to a brute-force attack. Asymmetric **encryption** schemes like RSA are subject to factoring attacks when used with a 512 bit modulus.

**HARDWARE AND SOFTWARE REQUIREMENT**

To run the program one must have:

A computer or a laptop with windows or Linux operating system installed.

Now to secure files/folder over the system, there is no requirement of hardware or software.

But for transferring files to other user.

If you plan to use the RSA option with DFSMSdss to encrypt the data-encrypting key, you must consider the cryptographic hardware that exists at the site that will decrypt the data. Not all types of RSA private keys are supported by all types of cryptographic hardware.

summarizes the RSA private tokens and required cryptographic hardware for decryption.

| RSA private key token (internal) | Required cryptographic hardware |
| --- | --- |
| RSA private key token 1024 — Modulus-Exponent Internal form | One of the following:   1. Cryptographic Coprocessor feature 2. PCI X Cryptographic Coprocessor 3. Crypto Express2 Coprocessor. |
| RSA private key token 1024 — Chinese Remainder Theorem Internal form | One of the following:   1. PCI Cryptographic Coprocessor 2. PCI X Cryptographic Coprocessor 3. Crypto Express2 Coprocessor. |
| RSA private key token 2048 — Chinese Remainder Theorem Internal form | One of the following:   1. PCI Cryptographic Coprocessor with LIC January 2005 or later, and z/OS® ICSF HCR770B or later 2. PCI X Cryptographic Coprocessor 3. Crypto Express2 Coprocessor. |
| Table 1. RSA private tokens and required cryptographic hardware for decryption | |

**SOFTWARE DEVELOPMENT**

Software development that is the technologies used in the making of project is PYTHON3 and knowledge of NETWORKING.

Since the project is all developed in python3 there is no other language or technology is required for networking. We have used various libraries for accessing system files, for encryption and decryption, for hashing passwords, and for transferring the data

whether it’s a file or simply a text message.

Some of libraries include:

os. pathlib

crypto.cipher crypto.hash crypto.Publickey crypto.random socket

threading tkinter

The design of the application is simple and user friendly which is designed with the help of tkinter.

Project is planned keeping in mind to provide the security to the user assets or data ( including each and every extension type of

file and folders). To ensure that all the content with in the system and while transferring to over the network is encrypted and safe.

So that if in transit any intruder try to intercept the data, the intruder will get only the encrypted data and is not able to fetch original data unless intruder has the decryption key. Even if your system is hacked by someone the hacker is not going to find any of your content if you have encrypted data with your preffered password.

**Python** is one of easiest languages, a language, which is **readable** and **understandable** by all developers (even devs who has zero experience with it). The idea of the **code examples** is to illustrate the crypto algorithms, encryption schemes and other cryptography concepts in action, not to promote certain library, API, language or technology.

Use the **code examples** as reference only, as guideline of how your code might look like, and adopt them to your favorite language and framework. Don't directly copy and paste the code examples. Sometimes we use a library, which is more user friendly and easier to install and use, instead of a faster and more reliable library from another vendor. Sometimes we use **hex-encoded** keys, ciphertexts, signatures and other values, in order to display them easier on the console, but in practice most apps will use binary encodings for increased performance and reduced network overhead.

You are free to adopt the code examples to other languages. At the end of the book we have given examples how to use cryptography in the most popular programming languages: **JavaScript**, **C#**, **Java** and **Python**. We skip giving examples with the lower level languages like **C** and **C++** because they work better for library writers (to implement efficiently certain algorithm), not for app developers. **C** and **C++** are more complex to setup and build, need more effort to manage the project dependencies and are more prone to errors.

Most **application developers** (e.g. Web devs, back-end devs, mobile devs and front-end devs) **use higher level languages** (like JS, Python, C#, Java, PHP and Go), but all of them will **understand the Python code** from the examples. If the code was given in C, it would be longer, more complex and less readable.

**Python is widely supported** everywhere (in Linux, Windows, Mac, on mobile devices and microcontrollers). All examples can also run online in a virtual machine in **online Python environments** such as: [**Repl.it**](https://repl.it/languages/python3) and [**PythonAnywhere**](https://www.pythonanywhere.com/)

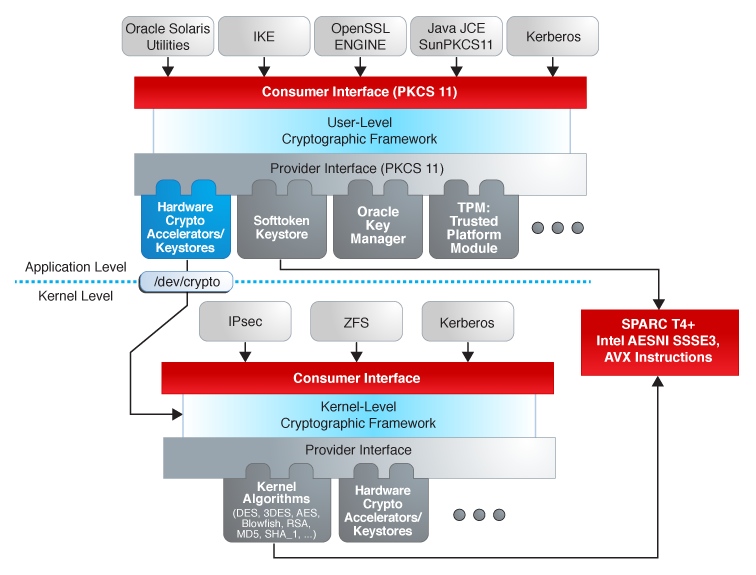
*Abstract:* *As the length of software keys increases to accommodate evolving needs for greater security, so the marketplace demands a wider variety of cryptographic implementations. With recent improvements in core design and frequency performance, designers are now asking whether customized IP blocks are still needed for these secure algorithms. In short, can a designer use a generic core in the hardware to save space and cost, and embed the cryptographic algorithms in software? The answer is simple—well, not so simple—it depends on the need.*

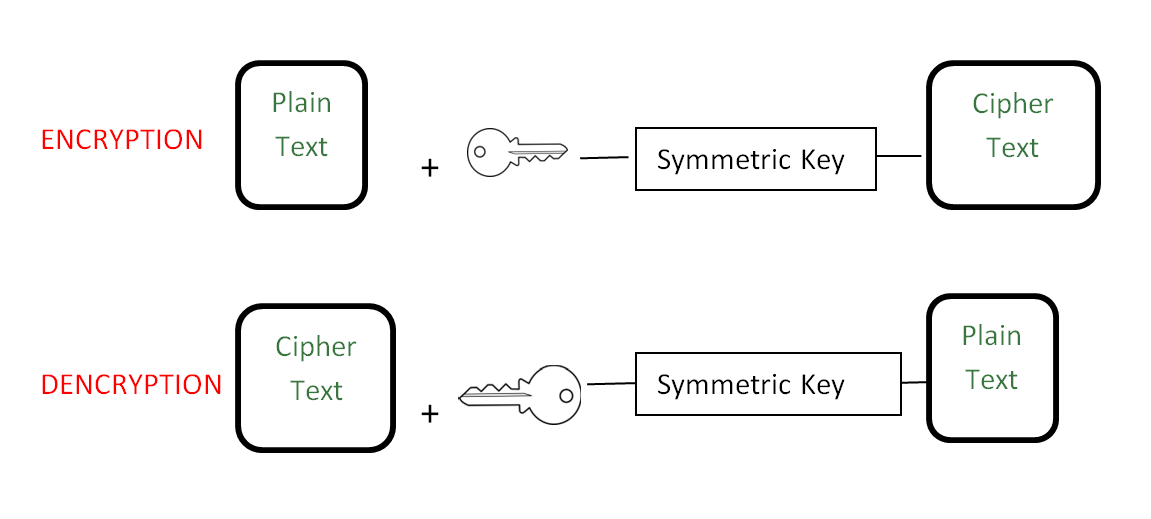
Cryptographic algorithms are high-performance, secure engines that require considerable space in a design. When countermeasures are added to thwart security attacks, the space and memory requirements grow even more demanding. For these reasons, cryptographic algorithms have traditionally been embedded as proprietary designs (i.e., intellectual property, IP) in hardware on smart cards or 8-bit chips. With recent improvements in core design and frequency performance, designers are now asking whether the customized IP blocks are still needed for these secure algorithms. In short, can a designer use a generic core in the hardware to save space and cost, and embed the cryptographic algorithms in software? The answer is simple—well, not so simple—it depends on the need.

**DESIGN FRAMEWORK:**

The Cryptographic Framework provides a common store of algorithms and PKCS #11 libraries to handle cryptographic requirements. The PKCS #11 libraries are implemented according to the RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki) standard.

**Figure1.** Cryptographic Framework Levels



****

### Concepts in the Cryptographic Framework

Note the following descriptions of concepts and corresponding examples that are useful when working with the Cryptographic Framework.

**Algorithms –** Cryptographic algorithms are established, recursive computational procedures that encrypt or hash input. Encryption algorithms can be symmetric or asymmetric. Symmetric algorithms use the same key for encryption and decryption. Asymmetric algorithms, which are used in public-key cryptography, require two keys. Hashing functions are also algorithms.

Examples of algorithms include:

* 1. Symmetric algorithms, such as AES
  2. Asymmetric algorithms, such as Diffie-Hellman and RSA
  3. Hashing functions, such as SHA256

**Consumers –** Users of the cryptographic services that come from providers. Consumers can be applications, end users, or kernel operations.

Examples of consumers include:

* 1. Applications, such as IKE
  2. End users, such as a regular user who runs the encrypt command
  3. Kernel operations, such as IPsec

**Keystore –** In the Cryptographic Framework, persistent storage for token objects, often used interchangeably with **token**. For information about a reserved keystore, see **Metaslot** in this list of definitions.

**Mechanism –** The application of a mode of an algorithm for a particular purpose.

For example, a DES mechanism that is applied to authentication, such as CKM\_DES\_MAC, is a separate mechanism from a DES mechanism that is applied to encryption, CKM\_DES\_CBC\_PAD.

**Metaslot –** A single slot that presents a union of the capabilities of other slots which are loaded in the framework. The metaslot eases the work of dealing with all of the capabilities of the providers that are available through the framework. When an application that uses the metaslot requests an operation, the metaslot determines which actual slot will perform the operation. Metaslot capabilities are configurable, but configuration is not required. The metaslot is on by default. For more information, see the [cryptoadm(1M)](http://www.oracle.com/pls/topic/lookup?ctx=E86824-01&id=REFMAN1Mcryptoadm-1m) man page.

The metaslot does not have its own keystore. Rather, the metaslot reserves the use of a keystore from one of the actual slots in the Cryptographic Framework. By default, the metaslot reserves the Sun Crypto Softtoken keystore. The keystore that is used by the metaslot is not shown as one of the available slots.

Users can specify an alternate keystore for metaslot by setting the environment variables ${METASLOT\_OBJECTSTORE\_SLOT} and ${METASLOT\_OBJECTSTORE\_TOKEN}, or by running the cryptoadm command. For more information, see the [libpkcs11(3LIB)](http://www.oracle.com/pls/topic/lookup?ctx=E86824-01&id=REFMAN3Flibpkcs11-3lib), [pkcs11\_softtoken(5)](http://www.oracle.com/pls/topic/lookup?ctx=E86824-01&id=REFMAN5pkcs11-softtoken-5), and [cryptoadm(1M)](http://www.oracle.com/pls/topic/lookup?ctx=E86824-01&id=REFMAN1Mcryptoadm-1m) man pages.

**Mode –** A version of a cryptographic algorithm. For example, CBC (Cipher Block Chaining) is a different mode from ECB (Electronic Code Book). The AES algorithm has modes such as CKM\_AES\_ECB and CKM\_AES\_CBC.

* **Policy –** The choice, by an administrator, of which mechanisms to make available for use. By default, all providers and all mechanisms are available for use. The enabling or disabling of any mechanism would be an application of policy. For examples of setting and applying policy, see [Administering the Cryptographic Framework](https://docs.oracle.com/cd/E53394_01/html/E54783/scftask-8.html#scrolltoc).
* **Providers –** Cryptographic services that consumers use. Providers plug in to the framework, and so are also called **plugins**.

Examples of providers include:

* + PKCS #11 libraries, such as /var/user/$USER/pkcs11\_softtoken.so
  + Modules of cryptographic algorithms, such as aes and arcfour
  + Device drivers and their associated hardware accelerators, such as the mca driver for the Sun Crypto Accelerator 6000
* **Slot –** An interface to one or more cryptographic devices. Each slot, which corresponds to a physical reader or other device interface, might contain a token. A token provides a logical view of a cryptographic device in the framework.
* **Token –** In a slot, a token provides a logical view of a cryptographic device in the framework.

### Cryptographic Framework Commands and Plugins

The framework provides commands for administrators, for users, and for developers who supply providers.

* Administrative commands – The cryptoadm command provides a –list subcommand to list the available providers and their capabilities. Regular users can run the cryptoadm list and the cryptoadm --help commands.

All other cryptoadm subcommands require you to assume a role that includes the Crypto Management rights profile, or to become superuser. Subcommands such as –disable, –install, and –uninstall are available for administering the framework. For more information, see the [cryptoadm(1M)](http://www.oracle.com/pls/topic/lookup?ctx=E86824-01&id=REFMAN1Mcryptoadm-1m) man page.

The svcadm command is used to manage the kcfd daemon and to refresh cryptographic policy in the kernel. For more information, see the [svcadm(1M)](http://www.oracle.com/pls/topic/lookup?ctx=E86824-01&id=REFMAN1Msvcadm-1m) man page.

* User-level commands – The digest and mac commands provide file integrity services. The encrypt and decrypt commands protect files from eavesdropping. To use these commands, see [Figure 2, Table 2, Protecting Files With the Cryptographic Framework Task Map](https://docs.oracle.com/cd/E53394_01/html/E54783/scftask-3.html#OSCMEscftask-24).

#### Administrative Commands in the Cryptographic Framework

The cryptoadm command administers a running Cryptographic Framework. The command is part of the Crypto Management rights profile. This profile can be assigned to a role for secure administration of the Cryptographic Framework. You use the cryptoadm command to do the following:

* Disable or enable provider mechanisms
* Disable or enable the metaslot

You use the svcadm command to enable, refresh, and disable the cryptographic services daemon, kcfd. This command is part of the Service Management Facility (SMF) feature of Oracle Solaris. svc:/system/cryptosvcs is the service instance for the Cryptographic Framework. For more information, see the [smf(5)](http://www.oracle.com/pls/topic/lookup?ctx=E86824-01&id=REFMAN5smf-5) and [svcadm(1M)](http://www.oracle.com/pls/topic/lookup?ctx=E86824-01&id=REFMAN1Msvcadm-1m) man pages.

#### User-Level Commands in the Cryptographic Framework

The Cryptographic Framework provides user-level commands to check the integrity of files, to encrypt files, and to decrypt files.

* digest command – Computes a message digest for one or more files or for stdin. A digest is useful for verifying the integrity of a file. SHA1 and SHA384 are examples of digest functions.
* mac command – Computes a [MAC](https://docs.oracle.com/cd/E53394_01/html/E54783/cryptogloss-2.html#OSCMEsecgloss-80) for one or more files or for stdin. A MAC associates data with an authenticated message. A MAC enables a receiver to verify that the message came from the sender and that the message has not been tampered with. The sha1\_mac and sha384\_hmac mechanisms can compute a MAC.
* encrypt command – Encrypts files or stdin with a symmetric cipher. The encrypt -l command lists the algorithms that are available. Mechanisms that are listed under a user-level library are available to the encrypt command. The framework provides AES, 3DES (Triple-DES), and Camellia mechanisms for user encryption.
* decrypt command – Decrypts files or stdin that were encrypted with the encrypt command. The decrypt command uses the identical key and mechanism that were used to encrypt the original file.
* elfsign command – Provides a means to sign providers to be used with the Cryptographic Framework. Typically, this command is run by the developer of a provider. The elfsign command has subcommands to request a certificate, sign binaries, and verify the signature on a binary. Unsigned binaries cannot be used by the Cryptographic Framework. Providers that have verifiable signed binaries can use the framework.

#### Plugins to the Cryptographic Framework

Third parties can plug their providers into the Cryptographic Framework. A third-party provider can be one of the following objects:

* PKCS #11 shared library
* Loadable kernel software module, such as an encryption algorithm, MAC function, or digest function
* Kernel device driver for a hardware accelerator

The objects from a provider must be signed with a certificate from Oracle. The certificate request is based on a private key that the third party selects, and a certificate that Oracle provides. The certificate request is sent to Oracle, which registers the third party and then issues the certificate. The third party then signs its provider object with the certificate from Oracle.

The loadable kernel software modules and the kernel device drivers for hardware accelerators must also register with the kernel. Registration is through the Cryptographic Framework SPI (service provider interface).

### Cryptographic Framework and Zones

The global zone and each non-global zone has its own system/cryptosvc service, which manages the Cryptographic Framework. When the cryptographic service is enabled or refreshed in the global zone, the kcfd daemon starts in the global zone, user-level policy for the global zone is set, and kernel policy for the system is set. When the service is enabled or refreshed in a non-global zone, the kcfd daemon starts in the zone, and user-level policy for the zone is set. Kernel policy was set by the global zone.

For more information about zones, see [*Introduction to Oracle Solaris Zones*](https://docs.oracle.com/cd/E53394_01/html/E54762/index.html). For more information about using SMF to manage persistent applications, see [Chapter 1, Introduction to the Service Management Facility in *Managing System Services in Oracle Solaris 11.3*](https://docs.oracle.com/cd/E53394_01/html/E54799/gmteb.html#SVSVFgmteb) and the [smf(5)](http://www.oracle.com/pls/topic/lookup?ctx=E86824-01&id=REFMAN5smf-5) man page.

### Cryptographic Sources and FIPS 140-2

FIPS 140-2 is a U.S. Government computer security standard for cryptography modules.

Oracle Solaris systems offer two providers of cryptographic algorithms that are approved for FIPS 140-2 Level 1.

* The Cryptographic Framework of Oracle Solaris is a provider of two FIPS 140-2 approved modules. The **userland** module supplies cryptography for applications that run in user space. The **kernel** module provides cryptography for kernel-level processes.
* The OpenSSL object module provides FIPS 140-2 approved cryptography for SunSSH and web applications. For information about the implementations of Secure Shell in Oracle Solaris, see [About Secure Shell in *Managing Secure Shell Access in Oracle Solaris 11.3*](https://docs.oracle.com/cd/E53394_01/html/E54793/sshuser-2.html#OSMSSsshuser-2).

Note the following key considerations:

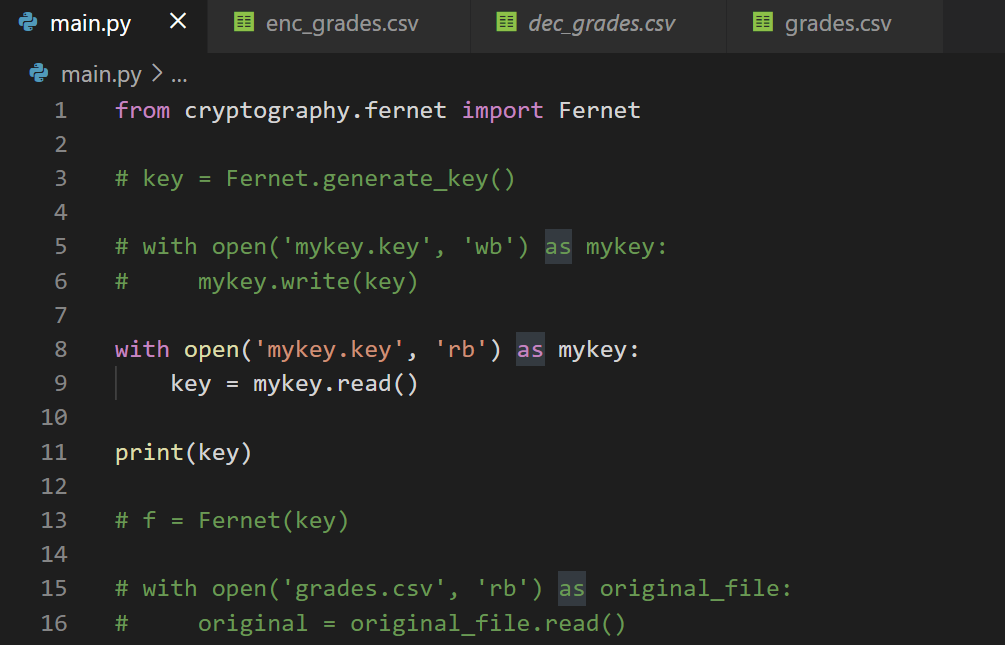
* Because FIPS 140-2 provider modules are CPU intensive, they are not enabled by default. As the system administrator, you are responsible for enabling the providers in FIPS 140-2 mode and configuring applications that use the FIPS 140-2 approved algorithms.
* If you have a strict requirement to use only FIPS 140-2 validated cryptography, you must be running the Oracle Solaris 11.3 SRU 5.6 release. Later Oracle Solaris releases build on this validated foundation and include software improvements that address performance, functionality, and reliability. Whenever possible, you should configure Oracle Solaris in FIPS 140-2 mode to take advantage of these improvements.

**IMPLEMENTATION**

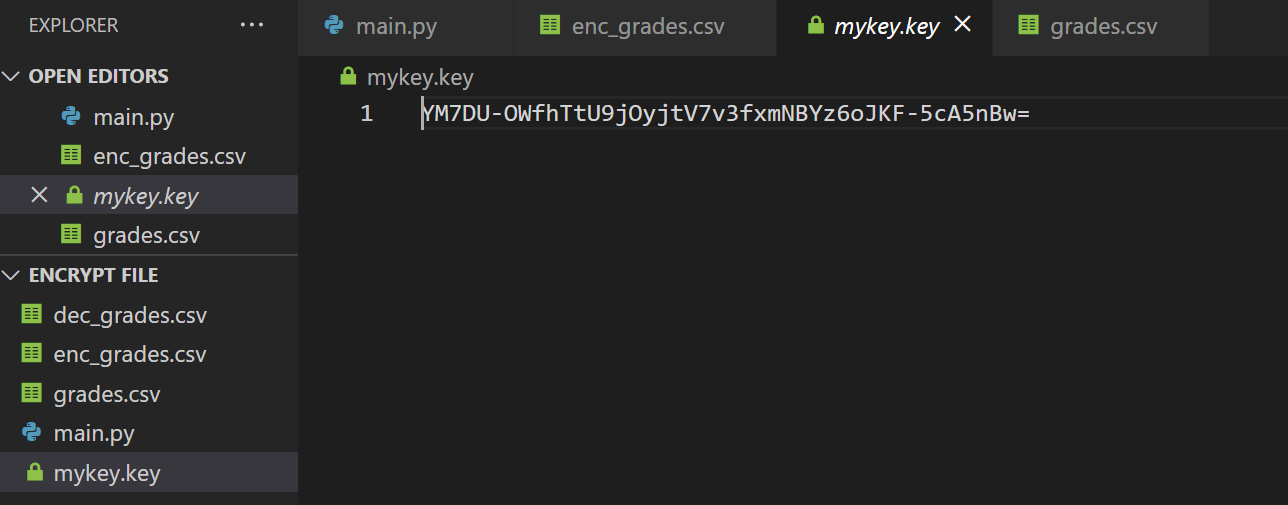
* 1. Grades Dataset



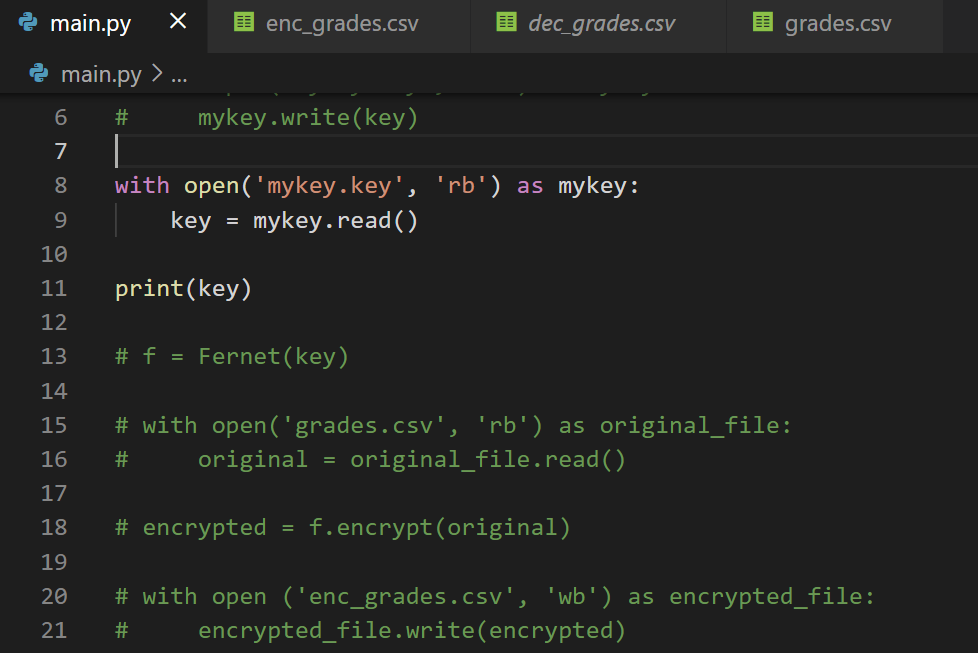
2. Generate Key Code



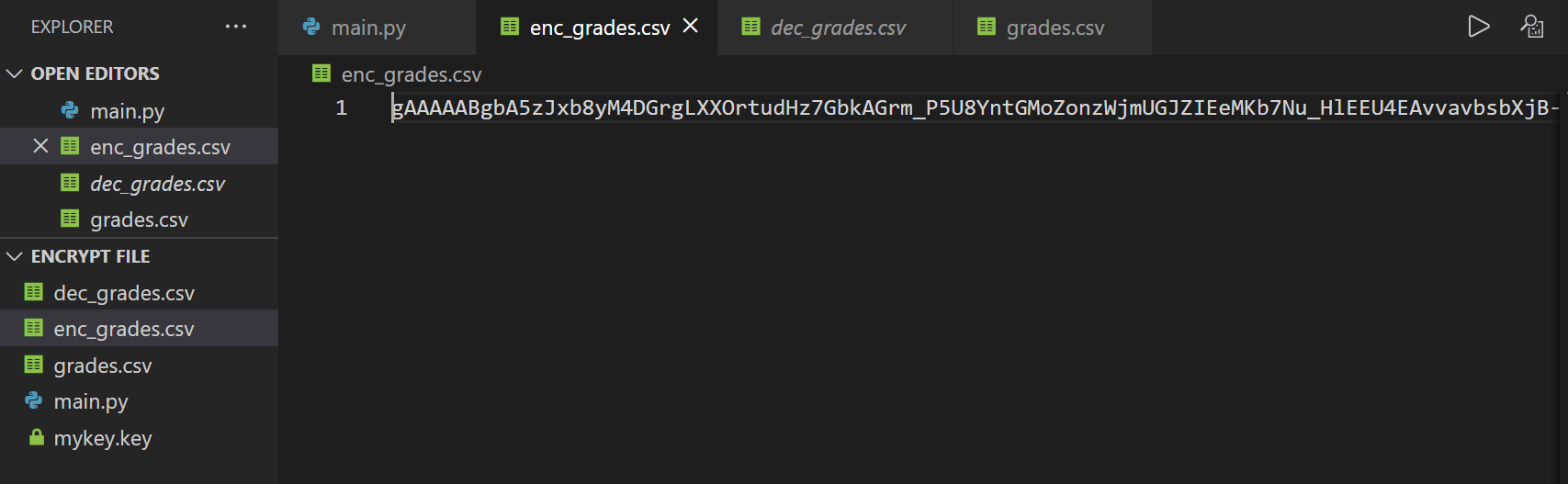
3. Generate Key Output



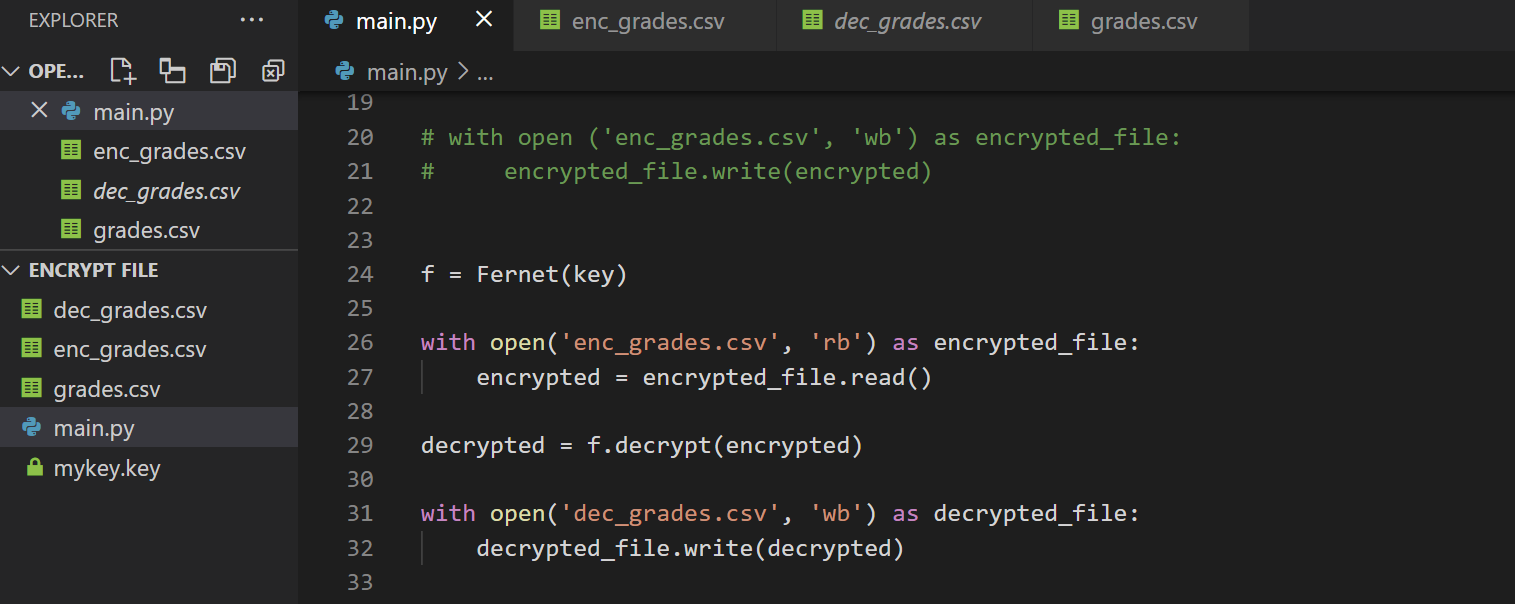
4. Encrypted Key Code



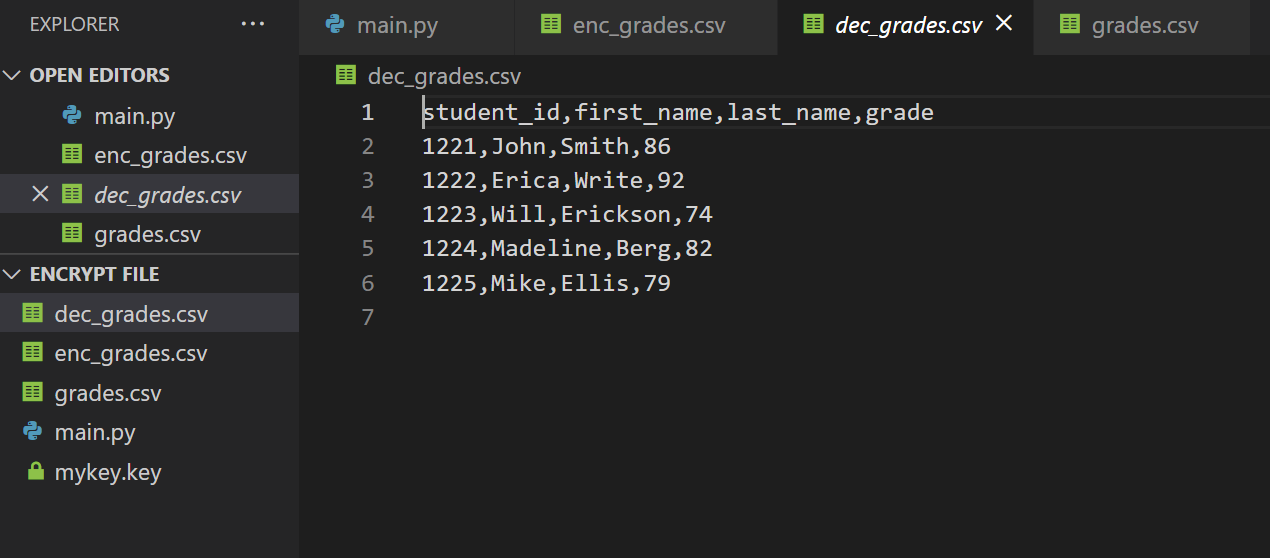
5. Encrypted Key Output



6. Decrypted Key Code



7. Decrypted Key Output



**TESTING PROCESS**

As of now the program is tested over the system for encryption and decryption of files and folder. For transferring files purpose the application is tested with in the local host and in the same newtork.

The application is not tested for transferring of files over the network.

**BENEFITS LIMITATION CONCLUSION AND FUTURE SCOPE**

BENEFITS:

Cryptography is an essential information security tool. It provides the four most basic services of information security −

* **Confidentiality** − Encryption technique can guard the information and communication from unauthorized revelation and access of information.
* **Authentication** − The cryptographic techniques such as MAC and digital signatures can protect information against spoofing and forgeries.
* **Data Integrity** − The cryptographic hash functions are playing vital role in assuring the users about the data integrity.
* **Non-repudiation** − The digital signature provides the non-repudiation service to guard against the dispute that may arise due to denial of passing message by the sender.

All these fundamental services offered by cryptography have enabled the conduct of business over the networks using the computer systems in extremely efficient and effective manner.

LIMITATION:

The limitation of the program is that if you are using share your file outside your network but you will be able to share with anyone within the same network.

Apart from the four fundamental elements of information security, there are other issues that affect the effective use of information −

* A strongly encrypted, authentic, and digitally signed information can be **difficult to access even for a legitimate user** at a crucial time of decision-making. The network or the computer system can be attacked and rendered non-functional by an intruder.
* **High availability,** one of the fundamental aspects of information security, cannot be ensured through the use of cryptography. Other methods are needed to guard against the threats such as denial of service or complete breakdown of information system.
* Another fundamental need of information security of **selective access control** also cannot be realized through the use of cryptography. Administrative controls and procedures are required to be exercised for the same.
* Cryptography does not guard against the vulnerabilities and **threats that emerge from the poor design of systems,** protocols, and procedures. These need to be fixed through proper design and setting up of a defensive infrastructure.
* Cryptography comes at cost. The cost is in terms of time and money −
  + Addition of cryptographic techniques in the information processing leads to delay.
  + The use of public key cryptography requires setting up and maintenance of public key infrastructure requiring the handsome financial budget.
* The security of cryptographic technique is based on the computational difficulty of mathematical problems. Any breakthrough in solving such mathematical problems or increasing the computing power can render a cryptographic technique vulnerable.

CONCLUSION:

The program is very useful in securing your assets/data (file/folder) over the system. For Company or institute where there is tire division the program is very useful.

Example: Suppose within a company if HR wants to send crucial information to his/her employee he/she can use this program and transfer files directly to each other without passing to any other employee within the company primises. Same for college if HODs want to communicate with particular faculty or any other in the staff.

One can also share files and folder over the network to anyone.

FUTURE SCOPE

**Elliptic Curve Cryptography** (ECC) has already been invented but its advantages and disadvantages are not yet fully understood. ECC allows to perform encryption and decryption in a drastically lesser time, thus allowing a higher amount of data to be passed with equal security. However, as other methods of encryption, ECC must also be tested and proven secure before it is accepted for governmental, commercial, and private use.

**Quantum computation** is the new phenomenon. While modern computers store data using a binary format called a "bit" in which a "1" or a "0" can be stored; a quantum computer stores data using a quantum superposition of multiple states. These multiple valued states are stored in "quantum bits" or "qubits". This allows the computation of numbers to be several orders of magnitude faster than traditional transistor processors.

To comprehend the power of quantum computer, consider RSA-640, a number with 193 digits, which can be factored by eighty 2.2GHz computers over the span of 5 months, one quantum computer would factor in less than 17 seconds. Numbers that would typically take billions of years to compute could only take a matter of hours or even minutes with a fully developed quantum computer.

In view of these facts, modern cryptography will have to look for computationally harder problems or devise completely new techniques of archiving the goals presently served by modern cryptography.

**REFERENCES**

<https://searchsecurity.techtarget.com/definition/cryptography>  
  
<https://docs.oracle.com/cd/E53394_01/html/E54783/scf-10.html>

<https://www.tutorialspoint.com/cryptography/cryptography_quick_guide.htm>

<https://www.maximintegrated.com/en/design/technical-documents/tutorials/5/5421.html>  
  
<https://www.electronicdesign.com/technologies/embedded-revolution/article/21132412/maxim-integrated-cryptographic-implementations-hardware-vs-software>  
  
<https://www.google.com/url?sa=t&source=web&rct=j&url=https://en.m.wikipedia.org/wiki/Cryptography&ved=2ahUKEwio_YSyscjwAhV_zjgGHZV0BH8QFjAhegQIVxAC&usg=AOvVaw3hbp94JBOc7EuruAX5tLos>